ECE 1778:
Creative Applications for Mobile Devices

Lecture 5

February 8, 2012

Today

1. Logistics
2. Proposal Discussions
3. Assignments P3 and A3
4. Programming Essentials for Android

Logistics

Assignments

- A3 and P3 are out today – will discuss later in lecture
  - A3 changed a little yesterday, in case you already acquired it before that
- A2 and P2 have been graded, with comments online
  - See blackboard portal
- General Comments on A2 from Braiden Brousseau (TA)
- General Comments on P2 from Daniel DiMatteo (TA)

Project TimeLine

1. Forming Groups
2. One-Page Proposal
   - Submitted Last Week – all are now approved
3. Project Plan
   - Due Today
4. **Proposal/Plan Presentations**
   - **Weeks of February 14 and 28 [No class in Reading Week]**
5. Spiral 2 & Spiral 4 Presentations
   - 2: March 6/13 4: March 20/27
6. Final Presentations
   - Weeks of April 3 & 10
7. Final Report Due April 12th

Project Plan Presentations

Plan Presentations Start Feb 14

- Formal Presentation
  - Using PowerPoint, Keynote, PDF or OpenOffice
- You will not know in advance if you're presenting on the 14th or 28th, to be fair, so come prepared to talk
- Send the presentation to me by email by **Tuesday Morning, February 14, at 9am**
  - jayar@eecg.utoronto.ca

Time Limit

- **Five Minute Time Limit**
  - I will start timer that makes annoying sound, and then ask for wind-down
  - Omit needless words
- **Three Minutes for Questions**
  - Can be extended if seems worthwhile

Presentation Contents

The Essence of the Plan submitted today:

1. Reprise Goal, make more precise
2. Give Mock-ups of What User Sees
3. Block Diagram of Code
4. Statement of Risks/Issues
5. What do you need to learn that you don't know
6. For Groups with Appers
   - 1 extra minute, for Apper to say how this fits into their field

Did I Mention the Five Minute Limit?

- This is both serious and important
  - To this course and your ability to communicate going forward
- How many slides can there be in 5 minutes?
- How much can go on a slide?
- Are pictures good things in presentations?

How Do You Know if Presentation is Good?

- Practice it, standing up, in front of:
  - First, no-one
  - Then, a few others
  - In your presentation, tell us how many times you practiced it
  - Not too much, either, shouldn't sound memorized

- **Time it**
  - if too long, cut it

How Do You Know if Presentation is Good?

In Practice:

- Listen to what you are saying
- Does it make sense **listening with the ears of the audience?**

Who is Your Audience?

- A mixture of technically-literate and people with expertise in some another area [different from your own!]
- Make sure the lay people know **what** you're doing - the goal
- OK to go somewhat technical after that, but don't assume we're all expert in every sub-field of ECE and CS

Proposal Discussions, continued

One Member from Each Group

- Please stand up, and describe your proposal

- What & Why
  - Describe the idea, and its motivation
- Scope
  - Give a good sense of functionality – what is involved
  - Show that you've thought about the pieces

- Apper: how it relates to field/expertise

Groups

- Need Names for Group Projects
- Need to check who has already spoken

Group #	Project Name	Apper	Apper Field	Programmer1	Field/Degree	Programmer2	Field/Degree	Platform
1		Marc Halatsis	iSchool	Felix Lazbin	ECE/M.Eng (?)	Blair Fort	ECE/PhD	Android
2		Adrian Matheson	Industrial Eng/PhD	Frances Awachie	ECE/?	Matthew Thorpe	ECE/M.Eng	Android
3		Justin Chee	Rehabilitation Science/PhD	Tuck-Voon How	IBBME/PhD	Eric Wan	ECE/M.A.Sc.	Android
4	Toronto Museum	Shannon Linde	iSchool	Abhinav Goyal	ECE/?	Sana Haghighi	ECE/M.Eng	Android
5		Scott Pollock	iSchool/Museum Studies	Xu Sheng	ECE/M.Eng	Tony Ming Zhou	ECE/M.Eng	iPhone
6	Learning Emotions	Alexandra Makos	OISE/PhD	Rebecca Dreezer	CS/PM	Cindy Lau	IBBME/M.A.Sc.	Android
7		Jill Cates	IMS/M.Sc.	Eddie/Zi Hi	ECE/M.Eng	Theodore Avery	CS/M.Sc.	iPad
8		Graham Candy	Anthropology/?	Steve Chun-Hao Hu	ECE/?	Chenliang Man	ECE/?	Android
9				Ani Tumanyan	CS/PM	Arsen Tumanyan	CS/PM	Android
10		Sam Liu	IMS	Jonathan Tomkun	IBBME/MHsc	Simran Fitzgerald	CS/PM	Android
10		Dario Kuzmanovic	Research Ethics/?	Valmiki Rampersad	ECE/M.A.Sc.	Colin Chung	ECE/M.Eng	Android
11				Heyse Li	MIE/?	Matthew Ma	CS/?	Android
12				Nanxsuan Wang	ECE/M.Eng	Peng Liang	ECE/M.Eng	Android
13	Robot Control			Fitsum Andargie	ECE/PhD	Paul Bovbel	MIE/PhD	Android
14				Mani Golafra	ECE/M.Eng	Ryne Yang	ECE/M.Eng	Android
15	AR Game			John Matienzo	ECE/M.Eng	Valentin Berbenetz	ECE/M.Eng	iPhone

New Assignments P3 and A3

Assignment P3

Location, Motion Sensors and Image Capture

- Learning how
  - to determine out where the phone is, geographically (GPS)
  - how it is moving (accelerometer)
  - how to use the camera to capture images

1. Read the relevant parts of
   - **Murphy** & Android Development site (Android)
     - Murphy doesn't have good coverage of sensors
     - This lecture has notes on sensors
   - or **Mark, Nutting & LaMarche** (iPhone)

Assignment P3 – for Programmers

- In response to being shaken,
  - phone takes a picture 1 second after the shaking stops, and
  - records the GPS location at the same time.
- Each location should be stored in a growing list;
  - when the user views the list item, your application should display the picture taken at that location.
  - The list should be maintained over separate invocations of the app (i.e. stored in a file)
  - it should be possible to delete a list item, which would remove the corresponding image in the file system.
- Due next week,
  - February 13th, 6pm, penalty for being late

Assignment A3 – for Appers

- Recall: one goal of course is to give experience in reaching across disciplines
- Anticipate that Appers will be teaching Programmers the **language** of their discipline, and the basic concepts
  - Please, programmers, ask questions – get jargon explained
  - **AND** vice-verse.

- Assignment A3 is an experiment in bringing Appers a little into the world of programming
  - To give you practice talking to each other

Basic Idea of A3

- Apper: Choose from one of 4 technical areas listed **that you are not already familiar with***, **that your programming partners are:**

  1. Searching and Databases
  2. Digital Signal Processing
  3. Optimization
  4. Internet Communication

* This wasn't in the original posted version of A3

Then – Part 2

- Spend an hour with your partners, learning about this area, and take notes.
  - Don’t use any other sources of information
- Write up those notes & submit this Friday (Feb 9, 6pm)
  - 500 words + pictures

Then – Part 3

- Pursue a deeper understanding of the topic, via Internet
- Write another 500 words, due February 13th, 6pm
  - Do a better job of describing the topic; add some nuance.
- Offer some additional commentary on your view of this learning process
  - how it went,
  - how much you learned from Part 2 vs. Part 3,
  - what would have made it better

Android Essentials

- Life Cycle
- Sensors
- Eclipse Debugging
- Pop-Up Messages – Toast & Alerts

Android Activity ‘Life Cycle’

Android Application Life Cycle

- Recall: Activities are screens that the user sees, and associated process
- Android manages these Activities as a **stack**.
- When a new activity is started, it is placed on the top of the stack and becomes the running activity

- The previous activity always remains below it in the stack,
  - and will not come to the foreground again until the new activity exits.

An Activity Can Be in 1 of 4 ‘States’

State 1: Active/Running

- Activity in the foreground of the screen (at the top of the stack)
- Has ‘focus’, meaning user interactions go to it.

State 2: Paused

- activity has lost focus but is still visible
- a new smaller or transparent activity has focus on top of the activity)
- A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.

Activity States 3 and 4

State 3: Stopped

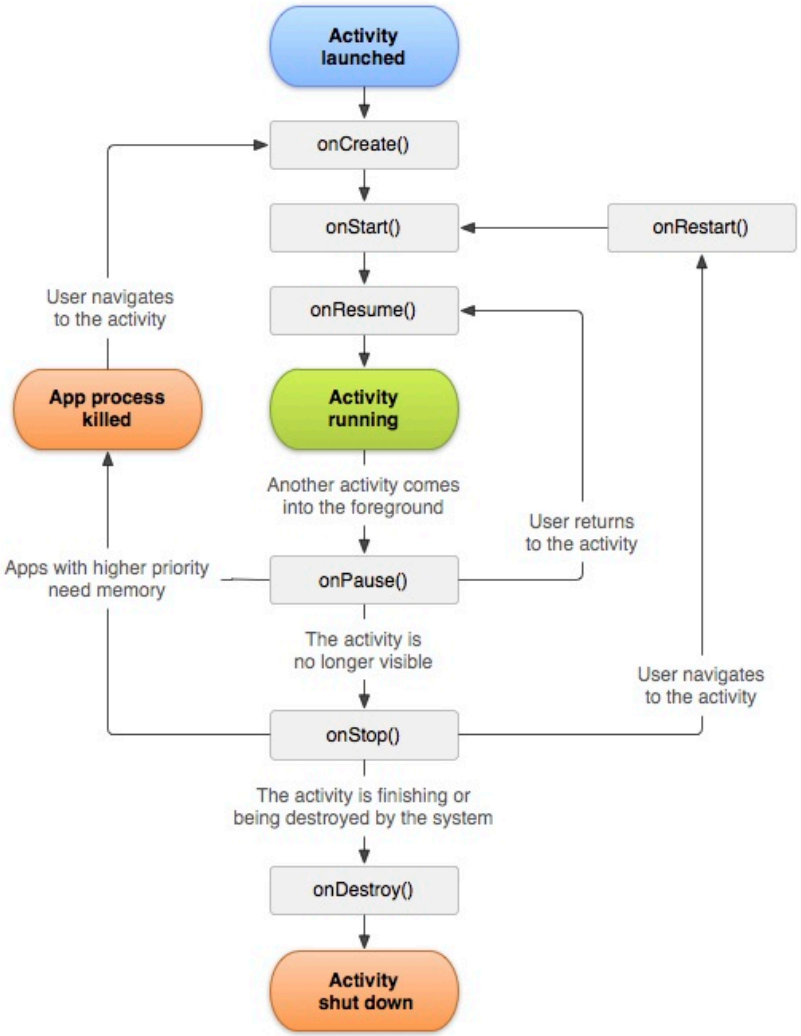
- activity is completely obscured by another activity
- retains all state and member information
- no longer visible to the user so its window is hidden
- it will often be killed by the system when memory is needed elsewhere.

State 4: Dead

- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, **or simply killing its process.**
- When displayed again to the user, it must be completely restarted and restored to its previous state.

Methods Called By Android to Change States

- Diagram shows states and methods that cause them to change
- Ovals: the 4 states
- Rectangles: methods called by android when it wants to change activity state



References

1. The Android Documentation:

http://developer.android.com/reference/android/app/Activity.html

2. Murphy, Busy Coder's Android, Chapter 18
   - "Handling Activity Lifecyle Events"

- Once your project gets going, it is really important to read through this and understand it
  - Last year's students pointed out that this was the key thing they had not understood in Android, that caused the most problems

The Key 'LifeCycle' Methods

OnCreate()

- Familiar with already – brings the activity to life

OnPause()

- Another Activity has gained the 'focus'
- Should stop any background threads, release large resources (such as a camera)
- **No guarantee that OnDestroy() will be called**, so best to save **all** state here

OnResume()

- Called as activity starts, **or** is restarted from a pause
- Can recall state from file, refresh the User Interface – see example

Key Loops in Life Cycle: 1 Entire Life

- The entire lifetime of an activity happens between the first call to onCreate(Bundle) through to a single final call to onDestroy().
- An activity will do all setup of "global" state in onCreate(), and release all remaining resources in onDestroy().
  - For example, if it has a thread running in the background to download data from the network, it should create that thread in onCreate() and then stop the thread in onDestroy().

2. Visible Lifetime

- The visible lifetime of an activity happens between a call to onStart() until a corresponding call to onStop().
- During this time the user can see the activity on-screen, though it may not be in the foreground and interacting with the user.
- Between these two methods you can maintain resources that are needed to show the activity to the user.
- For example, you can register a BroadcastReceiver in onStart() to monitor for changes that impact your UI, and unregister it in onStop() when the user an no longer see what you are displaying. The onStart() and onStop() methods can be called multiple times, as the activity becomes visible and hidden to the user.

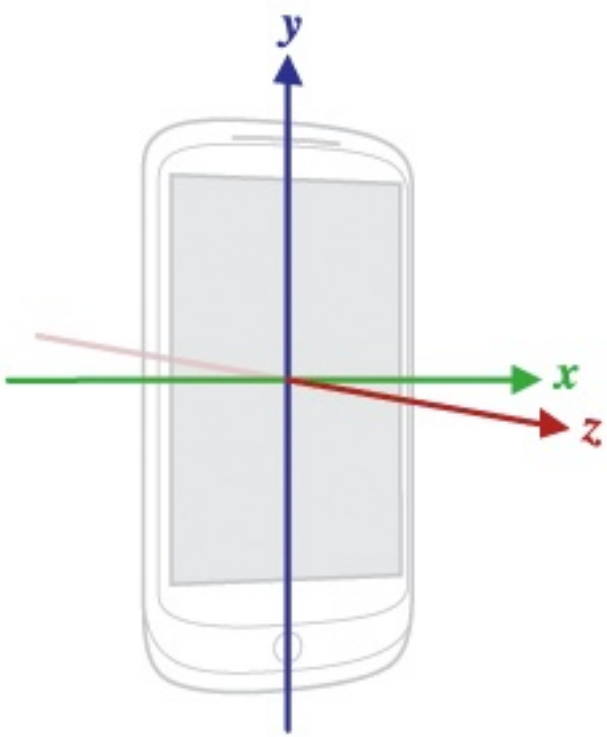
3. Foreground Lifetime

- The foreground lifetime of an activity happens between a call to onResume() until a corresponding call to onPause(). During this time the activity is in front of all other activities and interacting with the user.
- An activity can frequently go between the resumed and paused states -- for example when the device goes to sleep, when an activity result is delivered, when a new intent is delivered -- so the code in these methods should be fairly lightweight.

The Accelerometer

The Accelerometer

- The Accelerometer in phones is quite an exciting sensor!
- It can be used to feel motion in three dimensions
  - It samples the acceleration at least 100 times/second
  - It is very sensitive
- It measures acceleration in **m/s^2**
- Don't need permission to access sensors in manifest

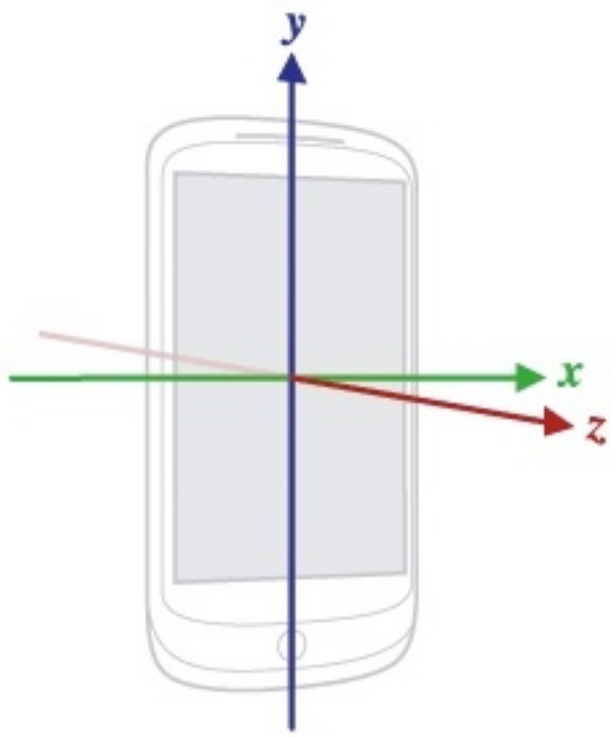


Gravity

- **Recall**: acceleration due to gravity is 9.8 m/s^2
- When the phone isn't moving, one or more of the axes will 'feel' this acceleration due to gravity or part of it.
- A phone that is falling, will feel 0 acceleration on all three axes

The Accelerometer Coordinate Space

- Coordinate-system is defined relative to the screen of the phone in its default orientation.
  - axes are **not** swapped when the device's screen orientation changes.
- X axis is horizontal & points right
- Y axis is vertical & points up
- Z axis points towards the outside of the front face of the screen.
  - coordinates behind the screen have negative Z values.



Example Application

- Read the accelerometers every time they change
- Output the raw values in each dimension
- Will include gravity

XML Layout – just a text field

<LinearLayout xmlns:android="*http://schemas.android.com/apk/res/android*"

....

<TextView android:text="*@+id/Output*"
android:id="*@+id/accText*"
android:layout_width="*wrap_content*"
android:layout_height="*wrap_content*">
android:textSize="*80sp*"</TextView>
</LinearLayout>

To Access a Sensor

Three key classes:

1. SensorManager
2. Sensor
3. SensorEvent

- Need to create a SensorManager class

SensorManager

- Is a class that
  - First lets you figure out what sensors are on the device
  - produces a list of all the sensors available,
  - Your program must need to check and see if the one you want is actually on the device:
- First, create the general sensor manager

```
myManager = (SensorManager) getSystemService
        (Context.SENSOR_SERVICE);
```

- Then, ask if the sensor you want is on the phone, e.g. the Accelerometer:

```
sensors = myManager.getSensorList
        (Sensor.TYPE_ACCELEROMETER);
```

SensorEvent Class

- Contains the sensor reading of the sensor, including:
  - Accuracy of the measurement
  - Sensor that generated the event
  - Timestamp
    - The time in nanoseconds at which the event happened
  - Values
    - e.g the three values of the acceleration along each axis

Front imports

```
package et.edu.aau.ece.jonathan.readaccel2;
import java.util.List;
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.util.Log;
import android.widget.TextView;
```


Declarations & usual

```
public class readaccel2 extends Activity {
    private TextView accText;
    private SensorManager myManager;
    private List<Sensor> sensors;
    private Sensor accSensor;
@Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
       accText = (TextView)findViewById(R.id.accText);
```

Set-up Sensor

```
// Set Sensor + Manager
myManager = (SensorManager)
getSystemService(Context.SENSOR_SERVICE);


sensors =
myManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
    if(sensors.size() > 0)
     {
        accSensor = sensors.get(0);
      }
  }
```

Listening, Reading, Outputing

```
private final SensorEventListener mySensorListener = new
SensorEventListener()  {
    public void onSensorChanged(SensorEvent event)  {
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];
        String output = String
         .format("x is: %f / y is: %f / z is: %f", x, y, z);
         accText.setText(output);  }
public void onAccuracyChanged(Sensor sensor, int
accuracy) {}};
```

Registering the Listener

```
@Override
    protected void onResume()
    {
     super.onResume();
     myManager.registerListener(mySensorListener,
     accSensor, SensorManager.SENSOR_DELAY_GAME);
    }
```

- Last parameter sets the frequency of updates

OnPause – Turn off accelerometer

```
@Override
    protected void onPause()
    {
     myManager.unregisterListener(mySensorListener);
     super.onStop();
    }
}
```

- i.e. unregister the listener

To Use the Compass (Magnetometer)

- Change:

```
sensors = myManager.getSensorList
                    (Sensor.TYPE_ACCELEROMETER);
```

- To:

```
sensors = myManager.getSensorList
                    (Sensor. TYPE_MAGNETIC_FIELD);
```

The Other Sensors

Useful Method – sensor maximum

- MaxRange = accSensor.getMaximumRange();
- Tells the largest value a sensor can deliver on a device

There are quite a few Sensors!

- Some are just different calculations with same sensor

1. Accelerometer:
   - Sensor.TYPE_ACCELEROMETER
2. Gravity
   - The accelerometer with non-gravity acceleration filtered out?
   - Sensor.TYPE_GRAVITY
   - Only available on Android 2.3 and later
3. Linear Acceleration
   - The accelerometer with gravity removed (filtered)
   - Sensor.TYPE_LINEAR_ACCELERATION
   - Only available on Android 2.3 and later

Compass/Magnetic field

- Sensor.TYPE_MAGNETIC_FIELD:
- Measures the magnetic field in three dimensions,
  - Measured in micro Tesla
- So, just Change:

```
sensors = myManager.getSensorList
                    (Sensor.TYPE_ACCELEROMETER);
```

- To:

```
sensors = myManager.getSensorList
                    (Sensor. TYPE_MAGNETIC_FIELD)
```

Gyroscope

- Sensor.TYPE_GYROSCOPE:
- Not on our phones
- Gives pitch, roll and yaw in radians/second
  - Measures motion more directly

Light Sensor

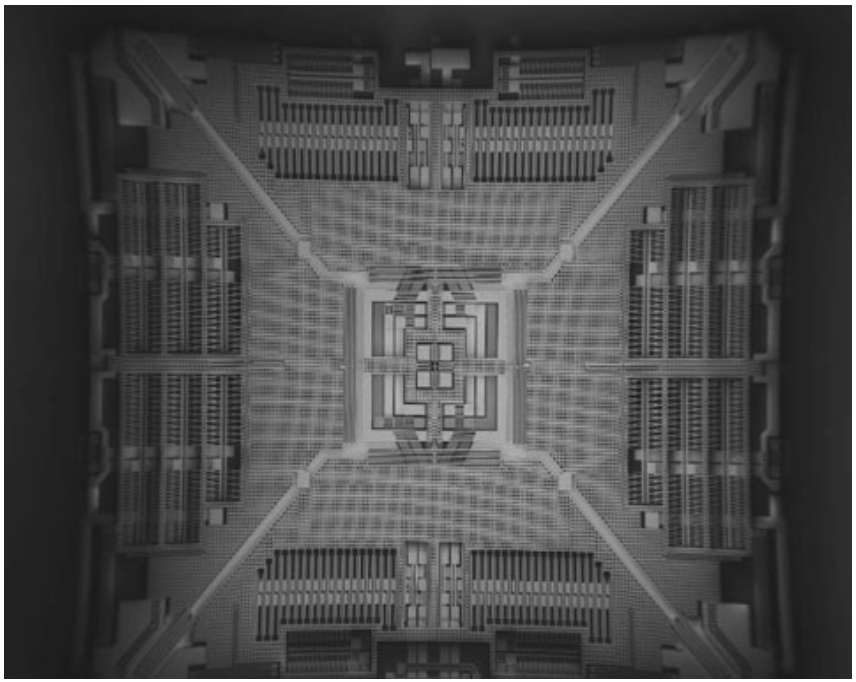
- Used for measuring ambient light to set screen brightness
- Measures the light, in Lux, coarsely
  - Seem like roughly 8 different values, maybe, when I tried it on the Nexus One
- Sensor. TYPE_LIGHT
- Available on Nexus S
- DEMO

Proximity Sensor

- Used for measuring how close the phone is to person's ear
- To turn off the touch screen
- Usually binary: **close** or **far**
- Sensor.TYPE_PROXIMITY

- DEMO

Sensors of The Future



Sensors to Come

- From the New York Times this past week:
- If you own a smartphone, you probably know just how smart those little gadgets are. They know where you're standing, the direction you're moving in and if you're holding the phone vertically or horizontally.
- To perform these clever tricks, the inside of your phone is stuffed with a number of sensors, which are little electronic components that can sometimes be as thin as a piece of paper.
- The coming generation of mobile phones — and other gadgets for that matter — will have so many new types of smart sensors that your current mobile phone will look pretty dumb.

Sensors to Come

- FromBenedetto Vigna, general manager of the MEMS division of STMicroelectronicsThe next smartphones would have altimeter sensors that would be able to detect your elevation. “
  - These sensors will tell people what floor they are on in a building, or could be used to more precisely determine where you are in relation to your friends on a location-based service,” he said.
- Other new sensors could include heart monitors to keep tabs on your health.

Sensors to Come

- There will also be sensors that can detect perspiration and could be used to monitor your excitement level and even mood.
- Additionally, phones will include more microphones, and temperature and humidity sensors to better determine their location and surroundings.

Sensors to Come

- This sensor-filled world will also affect video games. Sensors that can detect mood and excitement will usher in an era of video games that will factor in emotion during gameplay.
- Mr. Vigna said some technology companies were working on ways to increase security and privacy on mobile phones with sensors.
  - One way to do this is to build software that detects how you hold and interact with the device — almost like a motion fingerprint. After you use a new phone for a short period of time, it will start to learn your patterns and automatically lock or unlock the phone accordingly. This could be used for more secure banking too.

Sensors to Come

- David Pogue:
- As I've written in the past, it's only a matter of time before these sensors move beyond the smartphone and into people's clothes, glasses and homes.

- "Sensors will be everywhere in the next few years and will be able to help people become more conscious of the environment and our own health," explained Mr. Vigna. "Your socks, shoes, glasses and even your garbage can will have sensors inside designed to help you manage everything from your effects on the environment to your health."

Using the Debugger in Eclipse/Android

Debugging

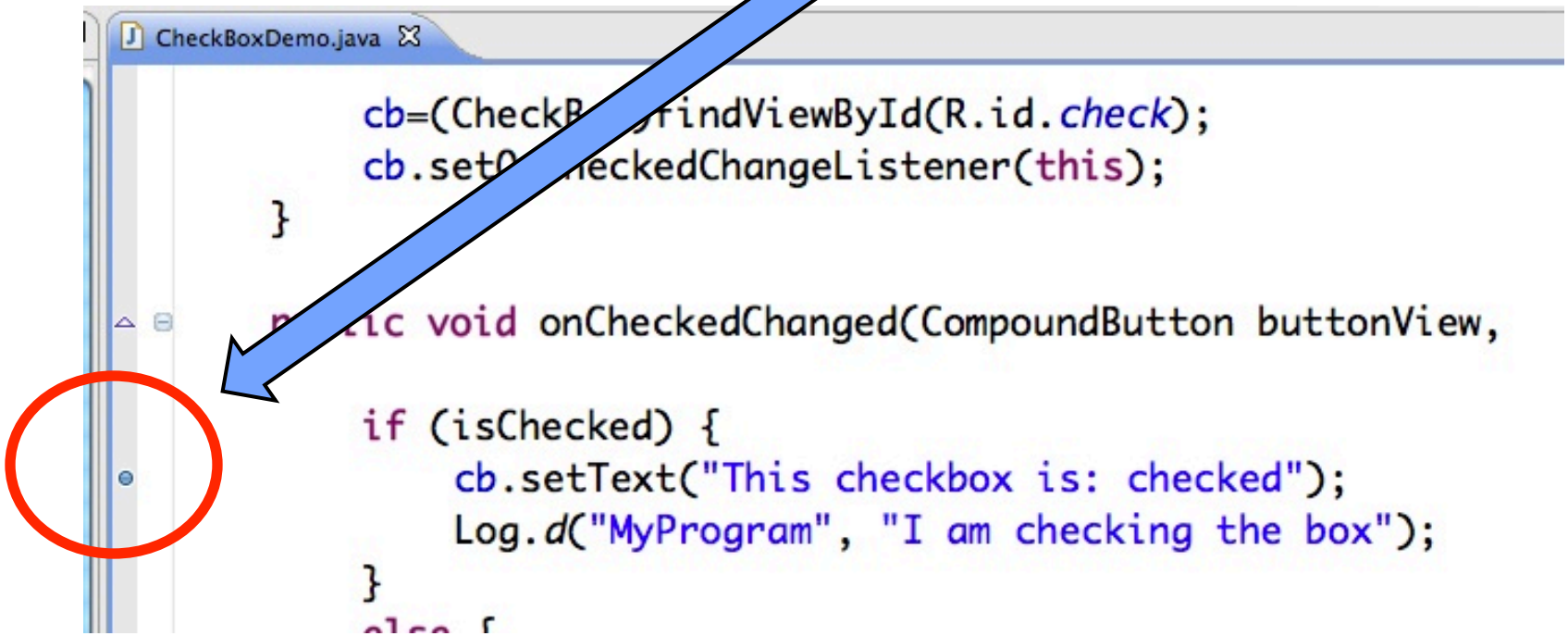
- Simple debug: use Log.d(TAG, “String”);
- Better: use the debugger in Eclipse/Android

- First, make sure your phone or emulator is set to enable ‘USB debugging’
- From the home screen:
  - Settings->Applications->Development
  - Check USB Debugging

Set a Breakpoint

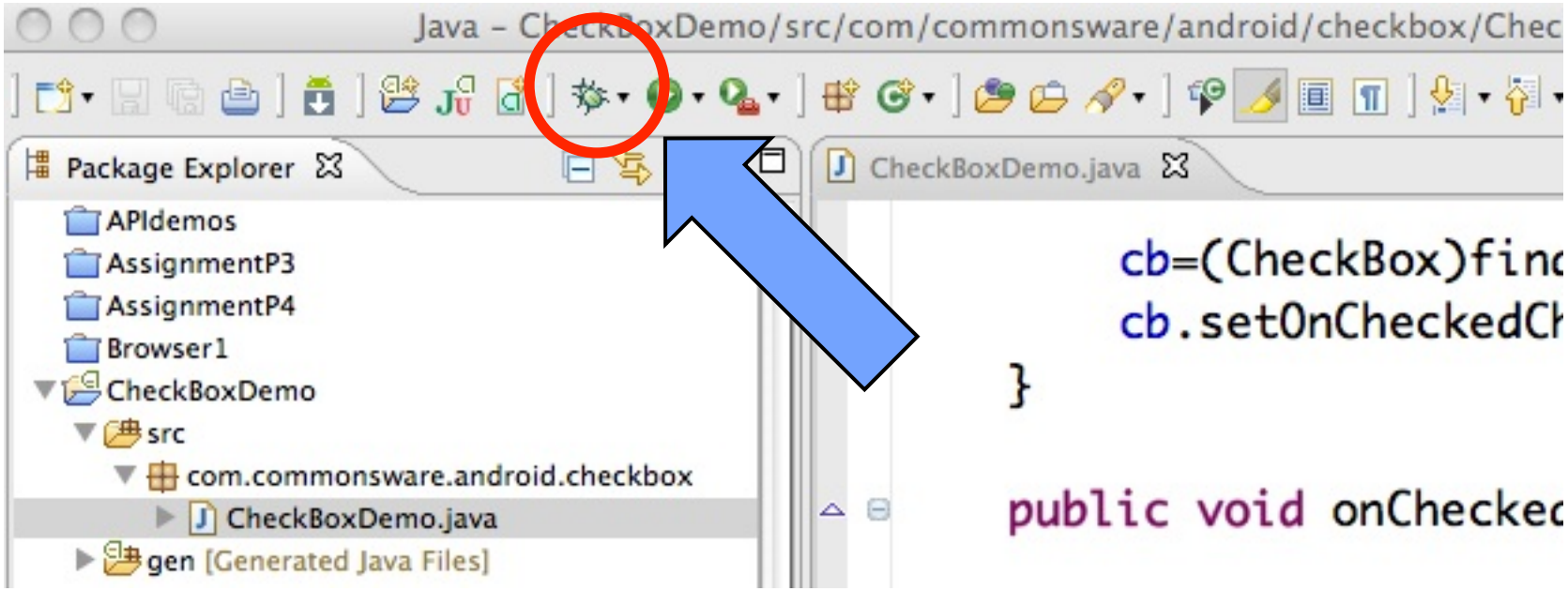
- Go to your source file,
- right click in the left-most column
- Select ‘Toggle Break Point’
  - Will set a breakpoint at that source code line (little blue dot)

Right Click over here



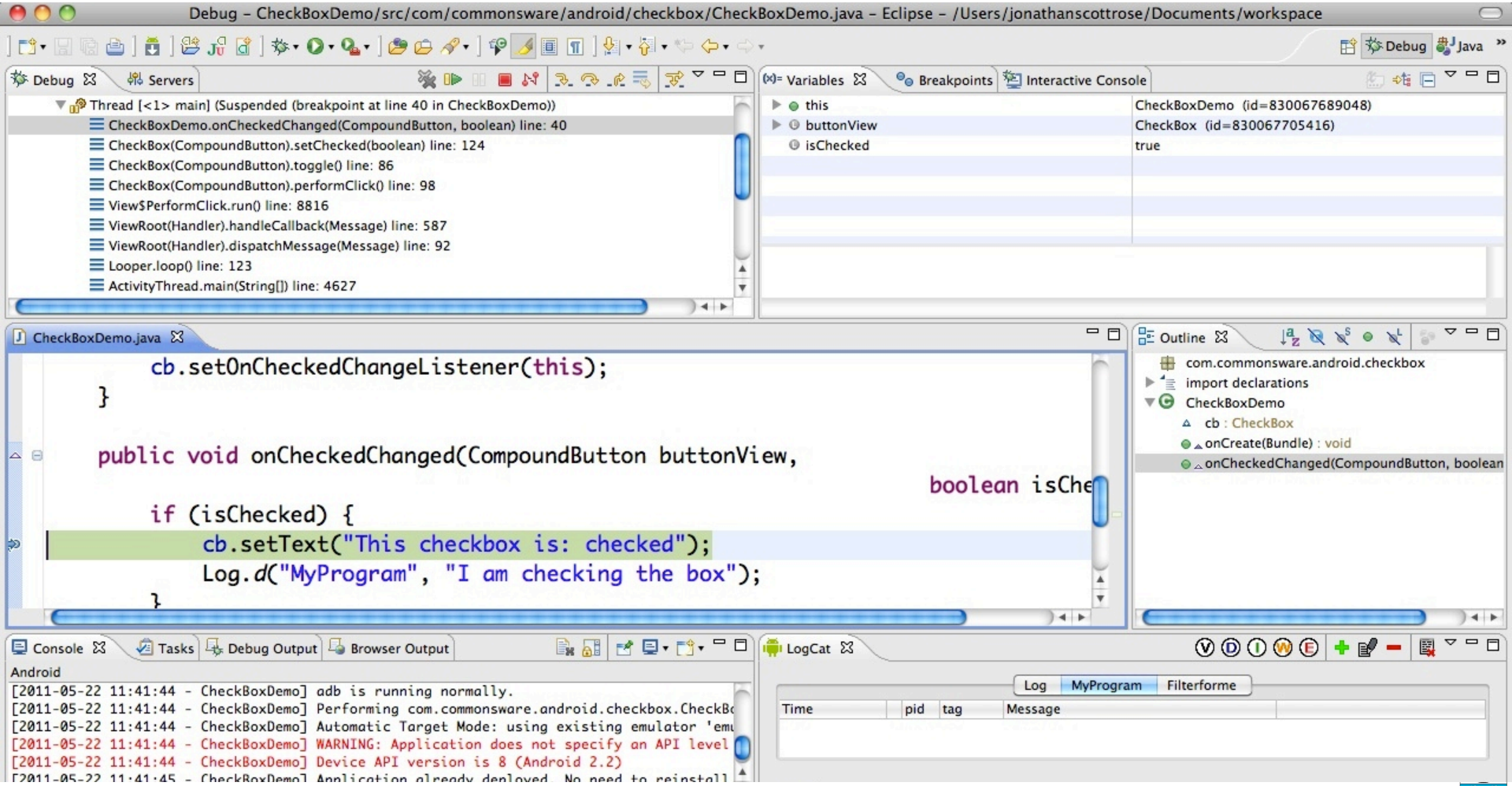
To Run the Debugger

- Click the ‘green bug’ to the left of to the ‘play’ button



Once the Breakpoint is Hit

- The whole Debugger perspective shows up:



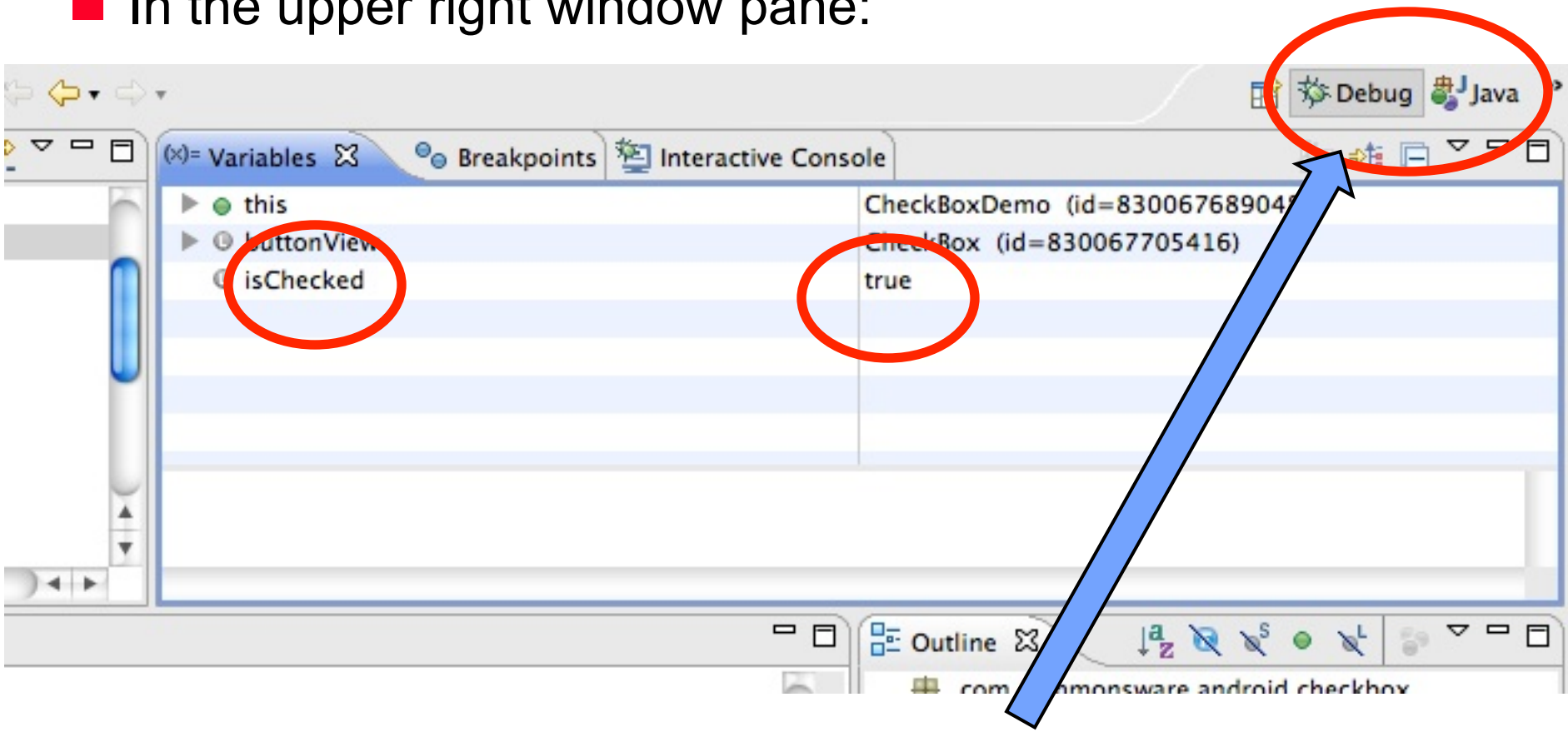
Can Single Step over, Step In, Run

- See this graphic near the top:

1. Continue running from Breakpoint
2. Step in to a function
3. Step over a line (single step)

Can View Value of Variables

- In the upper right window pane:

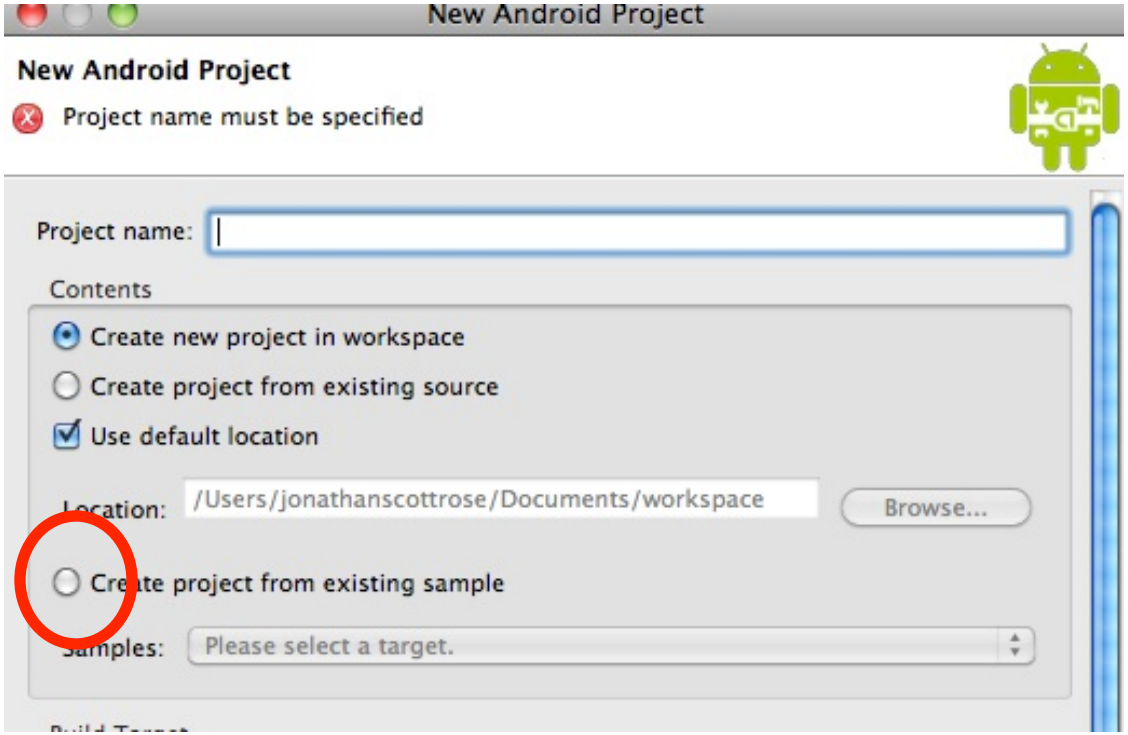


- To switch between regular and debug perspectives

API Demos!

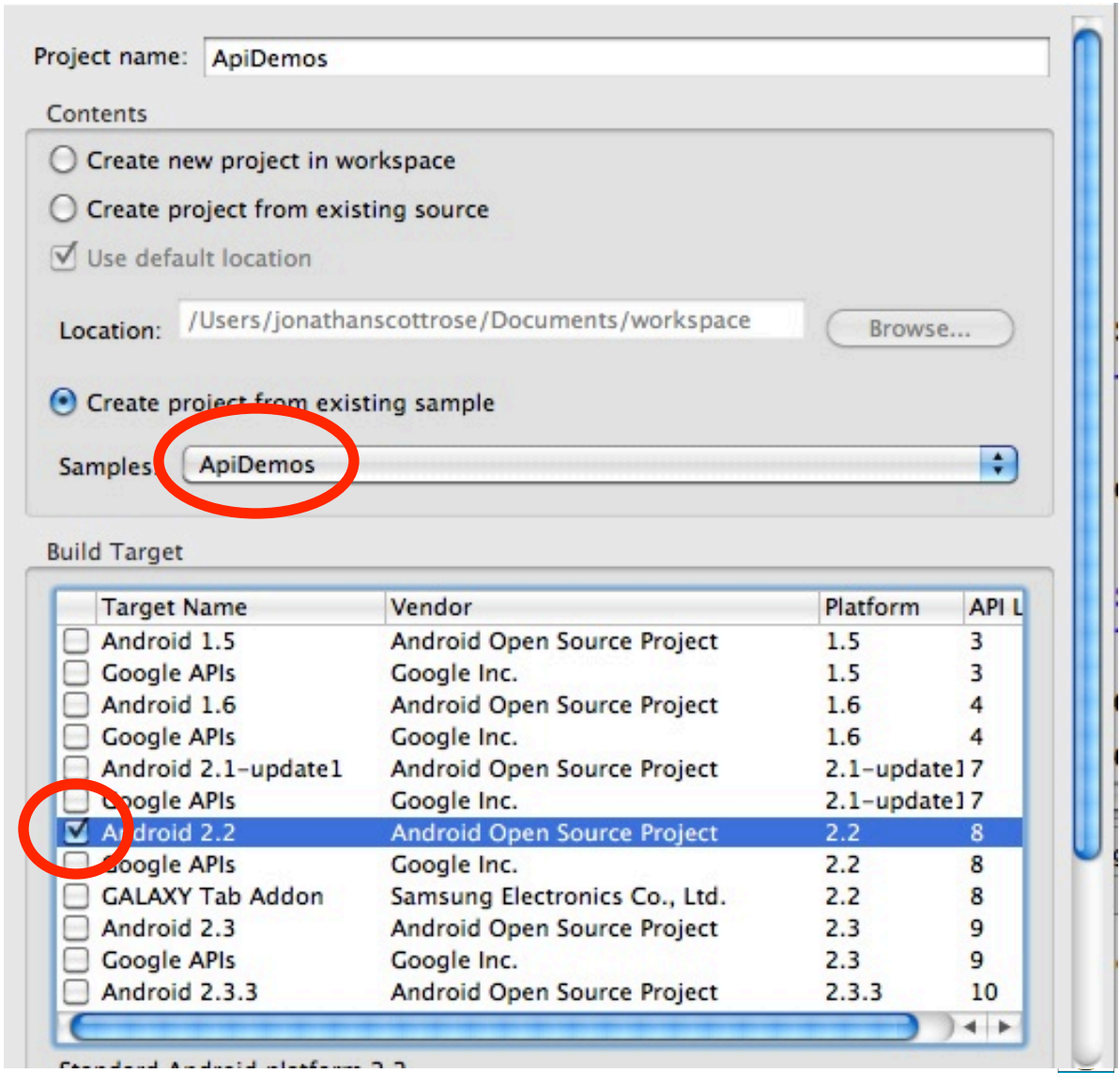
Google's API Demos

- Are great for learning everything Android
- Every feature of the phone is used, in a simple example
- To see them, create a new project in Android
- Select the button: "Create Project from existing source'



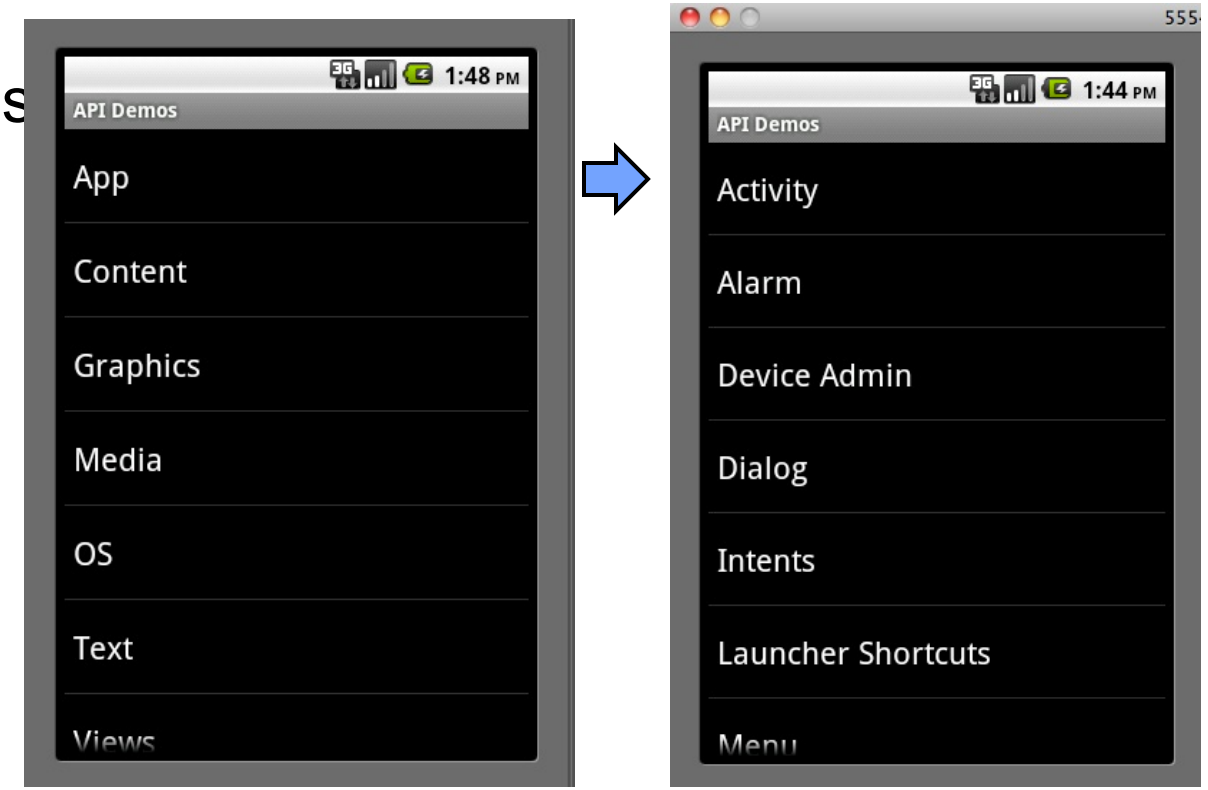
Then,

1. Select Android 2.2 as Build Target
2. Set Samples to ApiDemos
3. Finish



Then, Experiment and Learn!

- Each feature – activity, camera, sensor, is shown as a simple example, along with code
- Lots!
- Demo



Pop-up Messages

Pop-Up Messages

- Are really handy for conveying an alert or information to the user.

- Two kinds in Android:
  1. **Toasts**
  2. **Alerts**

Toasts

- A temporary message that appears and then disappears after a fixed amount of time
  - e.g. battery low warning
  - Someone signed in to Skype
  - Won lottery
- Purpose is to be un-obtrusive
  - Doesn't take 'focus' away from your activity

Toast Code – Easy!

- Create and show, all in one:

```
Toast.makeText(this,
                    "Eureka, you win!",
                    Toast.LENGTH_LONG).show();
```

Three fields:

1. 'this' is the current class context
   - Can also use **getApplicationContext**() instead
2. The text to be shown
3. Length of time to show
   - LENGTH_LONG or LENGTH_SHORT constants

Alerts

- A specific message that changes focus
- User must respond by clicking

- Three parts:
  - Message title
  - Actual message
  - Up to three response buttons:
    - Positive
    - Neutral
    - Negative
- Use OnClick to respond to buttons

Alert Code – using Toast to respond

```
new AlertDialog.Builder(this)
        .setTitle("MessageDemo")
        .setMessage("How are you feeling?")
        .setPositiveButton("I'm Positive",

 new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dlg, int sumthin) {
    Toast.makeText(MessageDemo.this, "Eureka, you win!",
                    Toast.LENGTH_LONG).show();
        }
})
```

Construction of the part of the Alert

A Toast in response to the positive click

Neutral Button in Alert

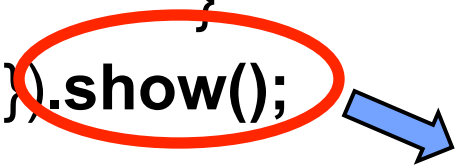
- A cascade of method calls

```
.setNeutralButton("Either Way",
    new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dlg, int sumthin) {
    Toast.makeText(MessageDemo.this, "Neutral Feeling",
                Toast.LENGTH_SHORT).show();
        }
})
```

Neutral Button in Alert

- Another cascade

```
.setNegativeButton("Either Way",
    new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dlg, int sumthin) {
   Toast.makeText(MessageDemo.this, ”I’m feeling bad",
                Toast.LENGTH_SHORT).show();
        }
}).show();
```



This shows the Alert

DEMO: MessageAlert